SAFEXPLAIN

Safe and Explainable
Critical Embedded Systems based on AI

Safe and explainable critical embedded systems based on AI

Francisco J. Cazorla

Barcelona Supercomputing Center (BSC)

HiPEAC 2023

MCS: International Workshop on Mixed Critical Systems – Safe and Secure Intelligent CPS and the development cycle

Workshop · Diamant (Level 1) · 10:00 - 17:00

Funded by the European Union

This project has received funding from the European Union's Horizon Europe programme under grant agreement number 101069595.

In a nutshell

- The scene
  - **Critical Embedded Systems (CES)** increasingly rely on Artificial Intelligence (AI): automotive, space, railway, avionics, etc.
  - CES must undergo **certification/qualification**
  - AI at odds with functional safety certification/qualification processes (**lack of explainability**, **lack of traceability**, **data-dependent** software, **stochastic** nature)
- SAFEXPLAIN ambition: architecting DL solutions **enabling certification/qualification**
  - Making them **explainable** and **traceable**
  - Preserving **high performance**
  - Tailoring solutions to varying safety requirements by means of **different safety patterns**

OCT 2022 – SEP 2025

SAFEXPLAIN

Safe and Explainable Critical Embedded Systems based on AI

BARCELONA SUPERCOMPUTING CENTER (BSC)
https://www.bsc.es/

IKERLAN, S. Coop (IKR)
https://www.ikerlan.es/

AIKO SRL (AIKO)
https://www.aikospace.com/

RISE RESEARCH INSTITUTES OF SWEDEN AB (RISE)
https://www.ri.se/

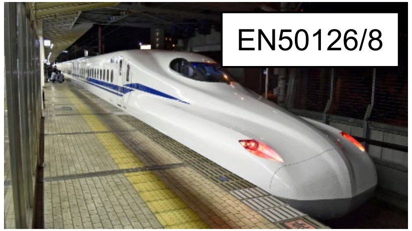
NAVINFO EUROPE BV (NAV)
https://www.navinfo.eu/

EXIDA DEVELOPMENT SRL (EXI)
https://www.exida-eu.com/

Jaume Abella
Project Coordinator

CES

- Failure or malfunction may result **severe harm** (casualties, economical loss)
- Exhaustive **Verification and Validation** (V&V) process, and **safety measures** deployed to guarantee the safety goals are met
- Each domain has it's own guidelines and regulations for SW and HW

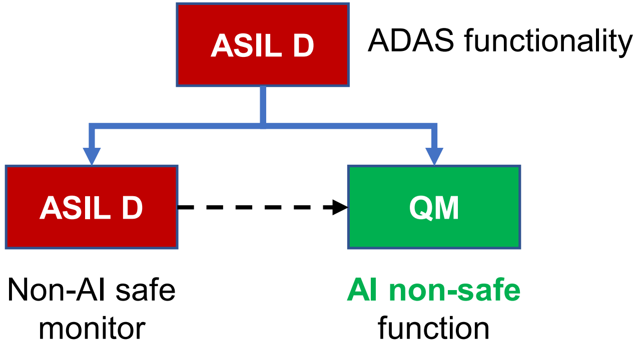


CES and AI

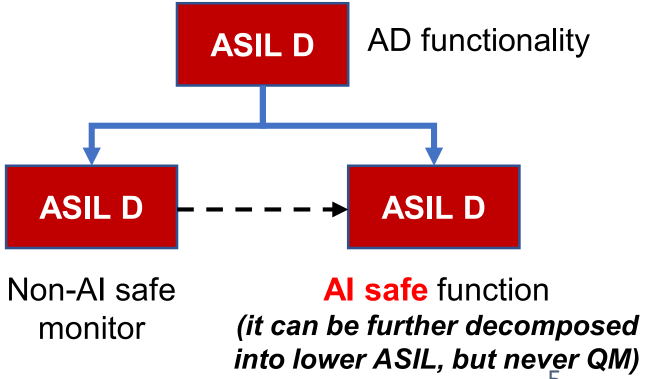
- The number of mechanical subsystems enhanced or completely replaced by electronic components is increasing
- Advanced software functions are becoming ubiquitous to control all aspects of CES, including safety related systems
- AI techniques, and Deep Learning (DL) in particular, are at the very heart of the realization of advanced software functions such as computer vision for object detection and tracking, path planning, driver-monitoring systems,…
- Autonomous operation
  - epitome of safety-related applications of AI in CES,
  - exemplifies the need for increasingly high computing performance whilst making AI solutions to comply with FUSA requirements

AI in Safety-critical systems so far and in the future

- When software/hardware implements safety-related functionality they inherit safety requirements
- Safety Integrity Level (SIL) decomposition
  - E.g., Automotive SIL (ASIL) from D (highest) to A (lowest), and then QM (no safety)
- **AI used in fail-safe systems** (i.e. systems with a safe state)
  - E.g., Advanced Driving Assistance Systems (ADAS) can notify misbehavior and transfer control to the driver

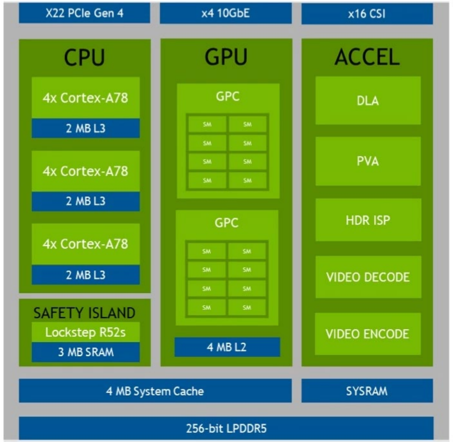


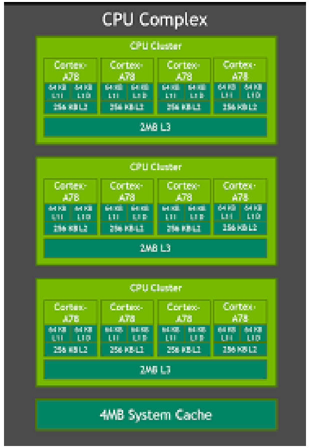
- With **autonomous systems** (cars, planes, satellites,…) this is **no longer doable**
  - No safe state available, hence AI components inherit safety requirements

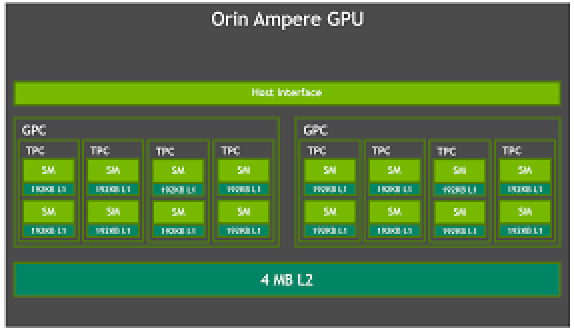


AI impact on the computing platform

- Software implements complex AI algorithms that manage huge amounts of data
- This carries huge computing performance requirements
- Hardware in safety-critical systems: from simple micro-controller to heterogeneous MPSoC with specific accelerators
- Complex MPSoC complicates established software timing V&V



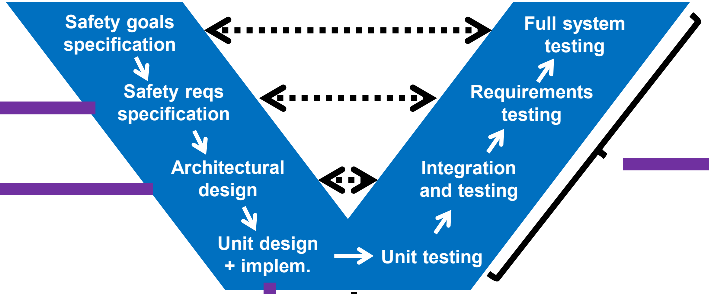




e.g. NVIDIA Orin
Source: NVIDIA

Safety-related Systems Development Process

- Usual V-model



Precise and **unambiguous requirements**

- HW: admits failure rates due to random faults
- **SW: fully deterministic. Cannot fail**

Control SW:
- Data is abstracted (name, source, but not its values)
- **Algorithm is "data-independent"** (defined independently of how the data looks like)

Code (SW) designed without a single experiment

Data used only for testing purposes: PASS/FAIL tests, diagnostics, quality assessment

But **data does not determine the design** of the system

NO DATA USED TO DESIGN THE SYSTEM

DATA USED ONLY FOR TESTING

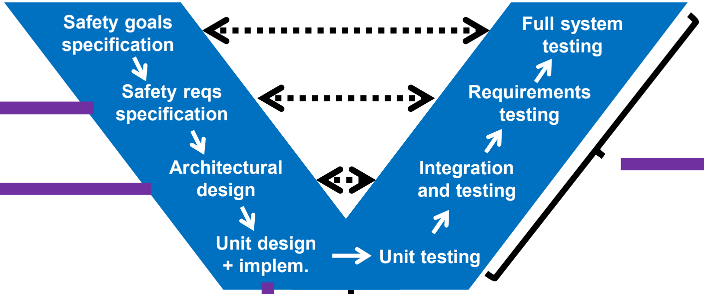
Safety-related Systems Development Process

- AI-related challenges

Precise and **unambiguous requirements**

- HW: admits failure rates due to random faults
- **SW: no longer deterministic. Can deliver erroneous output**

Data-defined SW:
- Algorithm **behavior is determined by actual data** (e.g., weights of a DNN)
- **Algorithm is fully "data-dependent"**



Code (SW) designed mostly based on experiments

Challenges to **define relevant data for testing**:
- Cannot be **exhaustive**
- **Equivalence** classes
- **Operation modes** cannot be enumerated deterministically
- Etc.

Moreover, **independence between training and test data** must be proven

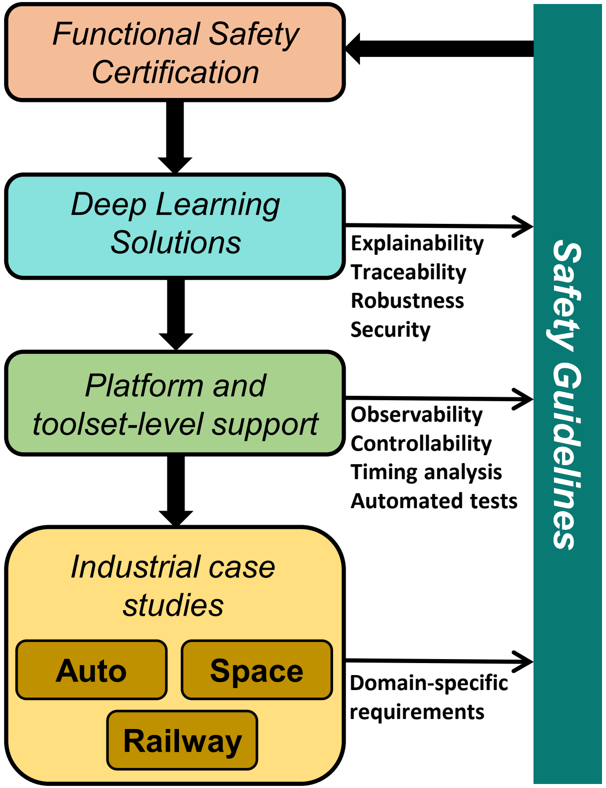
DATA DETERMINES SYSTEM DESIGN

AI (and DL) Specific Challenges

- Current practice in DL frontally clashes with Functional Safety (FUSA)-related processes since:
  - DL software is built as a **combination of**
    - **control** (model configuration such as what layers to use, in which order, etc.) and
    - **data** (algorithm parameters are obtained from training with specific datasets)
      - **stochastic nature**
      - **data-dependent nature**
  - There is a **lack of sufficient explainability and traceability**
    - Why each layer is used and what it does (**semantics**)
    - Why they are deployed in a specific order (**composed semantics**)
    - How safety **requirements can be traced** end-to-end
    - What the scope of application is (e.g. **valid input data range**)
    - What **confidence** can be reached on the predictions obtained (e.g. by detecting occlusions)
  - **Prediction accuracy is stochastic**, and test campaigns deliver, in the best case, success rates linked to specific testing datasets, therefore exposing to **dataset-dependent test conclusions** in many cases

Ambition/objectives

- Ambition: architecting DL solutions **enabling certification/qualification**
  - Making them **explainable** and **traceable**
  - Preserving **high and predictable performance**
  - Tailoring solutions to varying safety requirements by means of **different safety patterns**



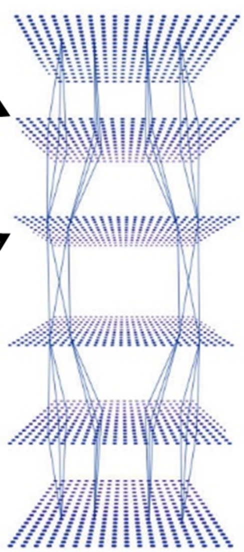
SAFEXPLAIN

SAFEXPLAIN Goals

- **GOAL 1**: Devise new DL components providing explainability and traceability by design

What **functionality** is performed by a given **layer**?

Why does this **layer** has these **characteristics** (e.g. shape, connections, etc.)?

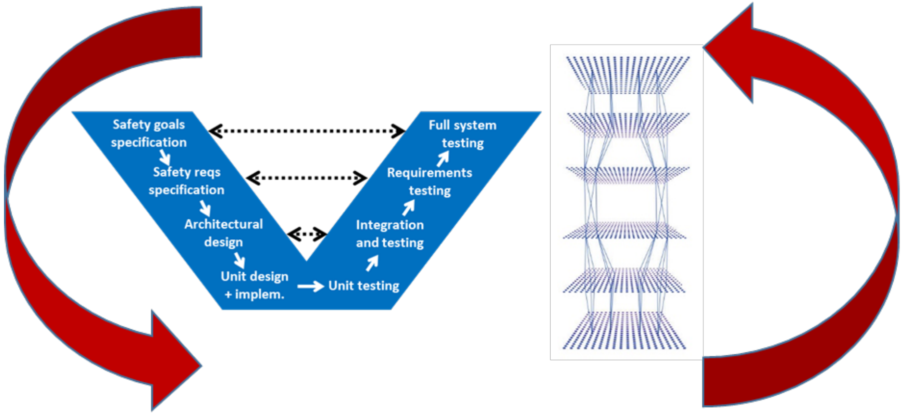


Why this **sequence** of **layers** and in this order?

Build DNNs so that we can explain
- **What each layer does**
- **Why each layer is as it is**
- **Why layers are arranged this way**
- **How requirements can be traced end-to-end**

SAFEXPLAIN Goals

- **GOAL 2**: Adapt software safety life cycle steps and the architecture of solutions based on DL components so that certification is viable

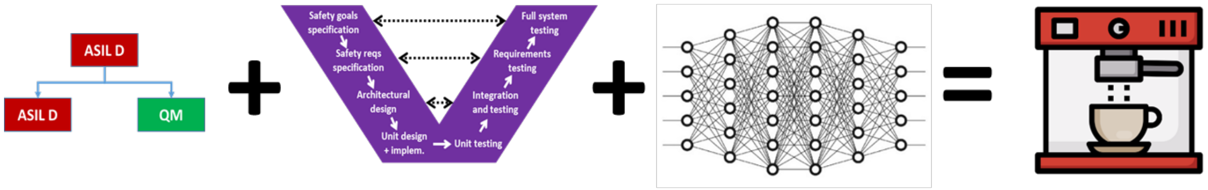


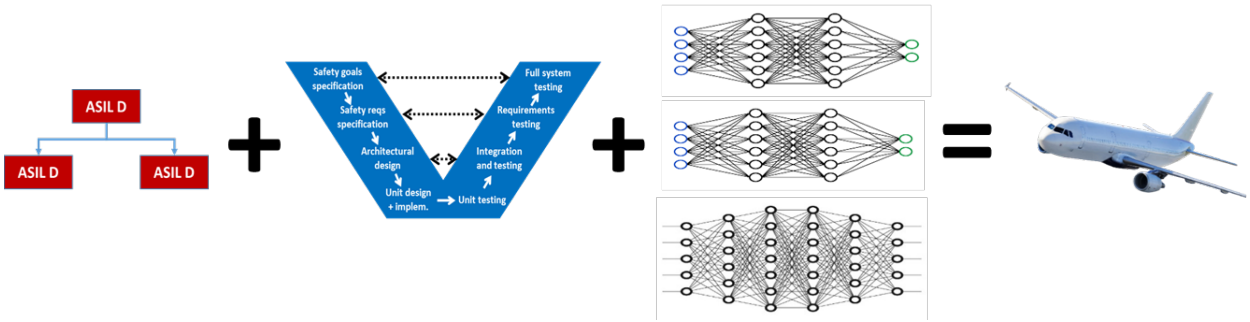
Tailor safety life cycle to enable DNN certification

Tailor DNNs to match properties needed by functional safety standards

SAFEXPLAIN Goals

- **GOAL 3**: Provide complementary safety patterns with different safety, cost, and reliability tradeoffs



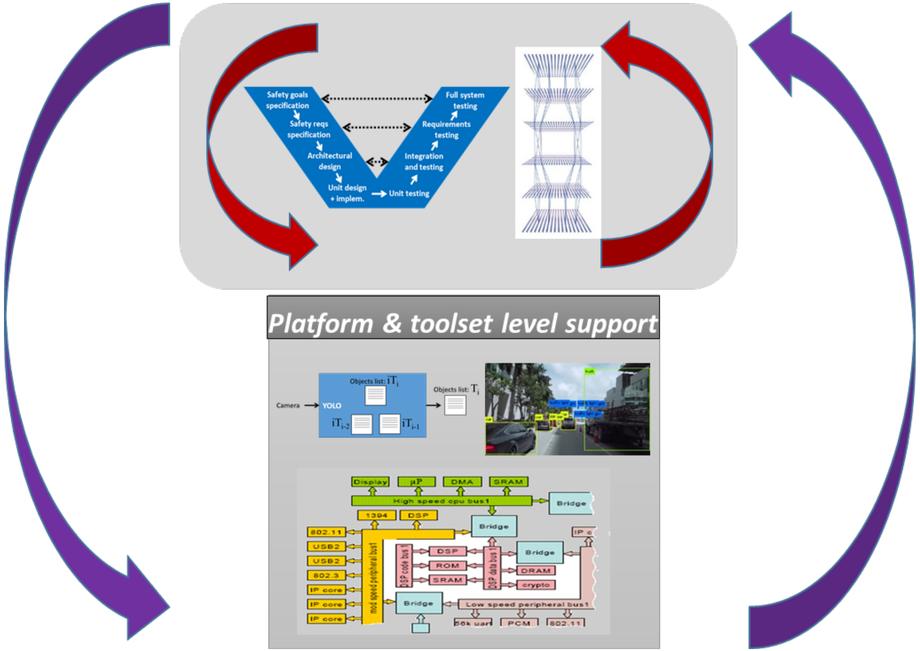


Safety patterns include:

- **SIL decomposition**
- **SIL allocation to DL items**
- **Development process**
- **DL architecture**
- **Etc.**

SAFEXPLAIN Goals

- **GOAL 4**: Tailor DL architectures to achieve sufficient performance on relevant high-performance platforms



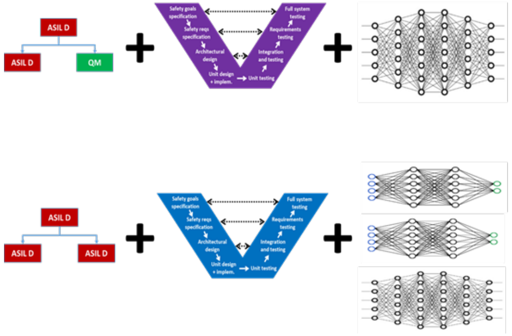
Coordinated actions:

- **Tailor DL architecture to the platform**
- **Keep DL architecture compatible with the safety life cycle**
- **Configure platform to achieve required performance**

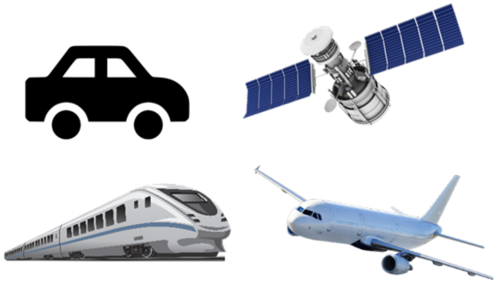
SAFEXPLAIN Goals

- **GOAL 5**: Demonstrate the long-term viability of the SAFEXPLAIN approach

SAFEXPLAIN technologies (safety patterns, DL libraries, etc.) → **Integrate in commercial and public tools** → **Assess against mixed-criticality industrial use cases**



- Open source DL libraries
- Commercial toolset integration



Putting it all together \1

- On the FUSA side
  - We must **identify patterns** (much preferably relevant cross-domains) meaningful for AI-based functions
  - Focus on **patterns with varying requirements** on AI-based functions
  - Identify **FUSA relevant properties** for DL components and ensembles

- On the DL side
  - Investigate **DL organizations** that make explainability and traceability emerge by construction while preserving accuracy
  - Investigate **combinations (ensembles) of DL models** that provide FUSA-relevant properties (e.g., diverse redundancy)

Putting it all together \2

- On the platform/tooling side
  - Consider DL solution deployments providing sufficiently **high and stable performance**
  - Iterate with FUSA and DL people to find FUSA patterns and DL solutions that can be run efficiently
  - Devise ways to (automatically or semi-automatically) **provide FUSA-relevant evidence** based on DL-based results using appropriate tools

- On the case study side
  - Consider **varying FUSA requirements** for different AI-based components
    - Within a single use case
    - Across different use cases
  - Consider heterogeneous requirements across use cases (e.g., **varying degrees of performance, accuracy**, etc.)

SAFEXPLAIN

Safe and Explainable
Critical Embedded Systems based on AI

Follow us on social media:

www.safexplain.eu

Funded by
the European Union

This project has received funding from the European Union's Horizon Europe programme under grant agreement number 101069595.

Safe and explainable critical embedded systems based on AI

Francisco J. Cazorla

Barcelona Supercomputing Center (BSC)

Funded by the European Union

This project has received funding from the European Union's Horizon Europe programme under grant agreement number 101069595.